Building Nice Command Line Interfaces

A Look Beyond The Standard Library

Europython 2015 - Bilbao

07 July, 2015

Patrick Mühlbauer

Software Developer @ Blue Yonder

blue yonder

CLI with Python - Where to start?

sys.argv?
getopt?
optparse or argparse?
Is there more?

blue yonder

Agenda

- $ **Click** - http://click.pocoo.org/
- $ **docopt** - http://docopt.org/
- $ **Cliff** - http://docs.openstack.org/developer/cliff/

blue yonder

Click

$ click_

- $ decorator approach
- $ highly configurable, good defaults

blue yonder

docopt

<docopt>

$ describe your CLI, get you parser

blue yonder

Cliff

- $ framework to create multi-level commands (something like git)
- $ setuptools entry points for subcommands
- $ output formatters

blue yonder

Minimal example - Click


```python
import click

@click.command()
def run():
    """Welcome to our brewery."""

if __name__ == '__main__':
    run()
```

blueyonder

Minimal example - docopt

<docopt>

```
"""Welcome to our brewery!

Usage:
    brewery [options]

Options:
    -h --help   Show this message and exit.

"""
from docopt import docopt

def run():
    args = docopt(__doc__)

if __name__ == '__main__':
    run()
```

blueyonder

Minimal example - Cliff


```python
import sys

from cliff.app import App
from cliff.commandmanager import CommandManager

class BreweryApp(App):
    def __init__(self):
        super(BreweryApp, self).__init__(
            description='Brewery demo app.',
            version='1.0',
            command_manager=CommandManager('cliff.brewery'),
        )

if __name__ == '__main__':
    brewery = BreweryApp()
    brewery.run(sys.argv[1:])
```

blueyonder

Subcommands - Click

$ click_

```python
import click

@click.group()
def run():
    """Welcome to our brewery."""

@run.command()
def list():
    """Show a list of all beers available in the brewery."""
    click.echo('Inside list command')

if __name__ == '__main__':
    run()
# python brewery.py list
# Inside list command
```

blueyonder

Subcommands - docopt

<docopt>

```
"""Usage: brewery [options] <command> [<args>...]

Options:
    -h, --help   Show this message and exit.

Commands:
    list         Show a list of all available beers.

"""
from docopt import docopt

def run():
    args = docopt(__doc__, options_first=True)

    if args['<command>'] == 'list':
        import brewery_list
        print(docopt(brewery_list.__doc__,
              argv=[args['<command>']] + args['<args>']))
```

blue yonder

Subcommands - Cliff


```
# setup.py
setup(
    # ...
    entry_points={
        'console_scripts': [
            'brewery_cliff = brewery:main'
        ],
        'cliff.brewery': [
            'list = brewery.commands:BeerListCommand',
            'buy = brewery.commands:BeerBuyCommand',
        ],
    },
    # ...
)
```

blue yonder

Subcommands - Cliff


```python
from cliff.command import Command

class BeerListCommand(Command):
    """Show a list of available beers."""

    def take_action(self, parsed_args):
        # code of list-command here
        # parsed_args: argparse.Namespace(arg1=3)
        pass
```

blue yonder

Options and arguments - Click

$ click_

```
import click

@click.group()
@click.option('--debug', is_flag=True)
def run(debug):
    """Welcome to our brewery."""
    if debug:
        click.echo('Running in debug mode.')

@run.command()
@click.argument('filter')
def list(filter):
    """List all beers of the brewery."""

if __name__ == '__main__':
    run()
```

blueyonder

Options and arguments - docopt

<docopt>

```
"""Welcome to our brewery!

Usage:
    brewery [options] <command> [<args>]...

Options:
    -h, --help  Show this message and exit.
    --debug     Run in DEBUG mode.

Commands:
    list        Show a list of all beers available in our brewery.

"""
```

blue yonder

Options and arguments - docopt

<docopt>

```
"""Usage: brewery list [options] [filter]

Options:
    -h, --help        Show this message and exit.

"""

# args = {'--help': False,
#         'filter': None}
```

blueyonder

Options and arguments - Cliff


```python
from cliff.command import Command

class BeerListCommand(Command):
    """Show a list of available beers."""

    def get_parser(self, prog_name):
        parser = super(BeerListCommand, self).get_parser(prog_name)
        parser.add_argument('filter')
        return parser

    def take_action(self, parsed_args):
        # code of list-command here
        pass
```

blueyonder

Repeating Arguments - Click

$ click_

```
@run.command()
@click.argument('filter', nargs=-1)
def list(filter):
    """List all beers of the brewery."""

# brewery list Pils Edelstoff
# -> filter = (u'Pils', u'Edelstoff')
```

blueyonder

Repeating Arguments - docopt

<docopt>

```
"""Usage: brewery list [options] [filter]...

Options:
    -h, --help        Show this message and exit.

"""
# brewery list Pils Edelstoff
# args = {'--help': False, 'filter': ['Pils', 'Edelstoff']}
```

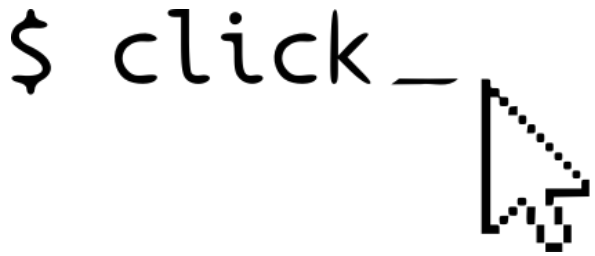
blue yonder

Repeating arguments - Cliff


```python
from cliff.command import Command

class BeerListCommand(Command):
    """Show a list of available beers."""

    def get_parser(self, prog_name):
        parser = super(BeerListCommand, self).get_parser(prog_name)
        parser.add_argument('filter', nargs='*')
        return parser

    def take_action(self, parsed_args):
        # code of list-command here
        pass
```

blueyonder

Defaults - Click




```python
@run.command()
@click.option('--name', required=True)
@click.option('--count', default=1)
def buy(name, count):
    """Buy COUNT bottles of NAME."""
```

blueyonder

Defaults - docopt

<docopt>

```
"""Usage: brewery buy [options] --name NAME [--count COUNT]

Options:
    -h, --help        Show this message and exit.
    --name NAME       The name of the beer to buy.  [required]
    --count COUNT     The number of bottles to buy.  [default: 1]

"""

# brewery buy --name Pils
# args = {'--count': '1', '--help': False, '--name': 'Pils'}
```

blue yonder

Defaults - Cliff


```
parser.add_argument('--name', metavar='NAME', required=True)
parser.add_argument('--count', metavar='COUNT', default=1)
```

blue yonder

Types - Click


```python
@run.command()
@click.option('--name')
@click.option('--count', default=1, type=click.IntRange(1, None))
def buy(name, count):
    """Buy COUNT bottles of NAME."""

# brewery buy --name Edelstoff --count 0
# Usage: brewery buy [OPTIONS]
#
# Error: Invalid value for "--count": 0 is smaller than
# the minimum valid value 1.
```

blue yonder

Types - docopt

<docopt>

Only strings and bools.
Typechecking has to be done by hand.

blue yonder

Types - Cliff


```
parser.add_argument('--name',
                    metavar='NAME', required=True)
parser.add_argument('--count',
                    metavar='COUNT', default=1, type=int)
```

blue yonder

ENVIRONMENT variables - Click

$ click_

```
@run.command()
@click.option('--name', default='Pils', envvar='BEER',
              help="Name of the beer you want to drink.")
def drink(name):
    """Drink a bottle of NAME. Cheers!"""
    click.echo("Drinking a refreshing cold {}.".format(name))

if __name__ == '__main__':
    run()

# python brewery.py drink
# Drinking a refreshing cold Pils.
# export BEER='Lagerbier Hell'
# python brewery.py drink
# Drinking a refreshing cold Lagerbier Hell.
```

blueyonder

ENVIRONMENT variables - docopt

<docopt>

Nope

blue yonder

ENVIRONMENT variables - Cliff


```
parser.add_argument('name',
                    default=os.environ.get('BEER', 'Pils'))
```

blue yonder

Testing - Click

$ click_

```python
from click.testing import CliRunner

def test_list_command():
    runner = CliRunner()
    result = runner.invoke(cli, ['list'])
    assert result.exit_code == 0
    assert 'Lagerbier Hell' in result.output
    assert 'stock: 1000' in result.output
```

blue yonder

Testing - Cliff


```python
# brewery.py
def main(argv=sys.argv[1:]):
    brewery = BreweryApp()
    return brewery.run(argv)

# test_brewery.py
# using pytests fixtures
def test_list_command(capsys):
    main(['list'])
    out, err = capsys.readouterr()
    assert 'Lagerbier Hell' in out
```

blue yonder

Cliff's List commands


```
from cliff.lister import Lister

class BeerLister(Lister):
    """Show a list of available beers."""

    def take_action(self, parsed_args):
        header = ('Beer', 'Alc.', 'Ingredients',
                  'Stock', 'Description')
        # ...
        # beer_rows has to be a tuple of tuples
        beer_rows = (('Pils', '5.6%', 'water, barley malt, hops',
                      242, 'Some description'),)
        return header, beer_rows
```

blue yonder

Summary

Click

- very robust
- many utilities

docopt

- flexible help screen creation
- implementations for lots of languages

Cliff

- output formatters very cool
- subcommand handling nice for plugins

blue yonder

We're passionate about Big Data. You too?

Then join us >>

www.blue-yonder.com